第2章

ARMコアの導入からシステムLSI設計，ソフト開発まで

――プロセッサIPのビジネス・モデルと設計手法

渡辺信久

ここではARMコアを実際に利用する手順や，ARMコアを取り巻くビジネス環境について解説する．ARM社は自身ではチップを販売せず，CPUコアの設計情報を他社にライセンスするIP（intellectual property）ベンダである．そのため，ARMコアを利用する際には，技術の知識に加えて，ライセンスなどの知識が必要になる．ARMコアに関する技術はARM社1社でサポートしているわけではなく，デザイン・センタやツール・ベンダなど，さまざまな業態の多数の企業によって支えられている．（編集部）

現在，英国ARM社のプロセッサ・コア（以下，ARMコア）は，32ビットCPUコアとして世界でもっとも多く使われています．ARM社が本格的にプロセッサIPビジネスに参入して10年ほど経ち，今ではいろいろな場面でその名まえを聞く機会も多くなっていることと思います．

ARM社は当初，「ライセンスを供与するのは一つの市場セグメントに1社」という戦略を採り，ライセンシ（ライセンスを受けた会社）がお互いに競合しないようにしていました．例えば，かつて総合家電メーカに勤めていた筆者が携帯電話用のCPUコアを探していたころ，同社に接触したのですが，すでにその分野でライセンシがいたため，断られた経験があります．その後2年ほどして，広く世界中の企業にライセンスを行う方針に変わったため，ARMコアのライセンシは急拡大していきました（右掲のコラム「組み込み機器への本格対応が遅れていた32ビットRISC CPU」を参照）．

さて，ARMコアには実際にどのようなものがあり，どのように入手できて，どのようにシステムLSI（SOC：system on a chip）へ組み込んでいくのでしょうか．こういったことは，半導体メーカなどでARM関連業務に就いている一部のエンジニアを除いて，わかりにくいのが現状かと思います．

本稿ではARMコアをシステムLSIに集積する際の全体像（ライセンスやビジネス・モデル，開発環境など）を紹介します．ARMコアを利用してみたいと考えている方のしごとの一助になれば幸いです．

1．ARMコアのライセンスと事業モデル

ARMコアの導入からシステムLSI設計までの一般的な手順は，図1のとおりです．

現在ライセンスされているARMコアには以下のファミリがあります．それぞれのファミリにはいくつかの派生タイプがあります．

- **ARM7ファミリ**――低消費電力で小面積のCPUコアとして，ARMコアが知られる先駆けとなったコア．もともと32ビット単一語長のRISCだったが，後にThumbと呼ばれる16ビット命令セットを追加して，コード効率を向上させた．
- **ARM9ファミリ**――DSP命令の追加や，Javaのハードウェア実行支援などの拡張を行っている．
- **ARM10ファミリ**――高性能化を目指してパイプライン段数がより多段になっている．
- **ARM11ファミリ**――16ビット命令セットをさらに強化

KeyWord　ARM，RISC，ライセンス，IP，ロイヤリティ，ESL，MaxSim，AMBA，サインオフ，トレードオフ

Column 組み込み機器への本格対応が遅れていた32ビットRISC CPU

ARMコアの先駆けとなったプロセッサは，1980年代前半のRISC技術が脚光を浴び始めたころ，すなわちMIPSやSPARCといったアーキテクチャとほぼ同時期に設計されていました．MIPSやSPARCは，エンジニアリング・ワークステーションをおもな用途として設計されていましたが，ARMプロセッサも当初は組み込み機器向けに開発されたものではありませんでした．ただ，設計リソースや資金面の制約により，とにかく低コスト，低ゲート数のCPUを設計する必要に迫られていました．そのため，汎用レジスタ数は16個と，ほかのRISCプロセッサと比べて少なくなっています．MIPSやSPARCと大きく異なる点は，ARMが，ほぼすべての命令で条件フラグの状態によって実行の可否判断を行えることです．また，シフト＋演算，マルチデータのロード/ストアを1命令で行える，特徴のある命令セットになっています．

ARM，MIPS，SPARCに共通するところは，制御フラグ操作を多用する組み込み機器制御にとくに必要とされるビット操作命令やブーリアン・ビット操作命令，テーブル参照命令，ベクタ割り込みなどがなかったことです．これらの命令や割り込み方式は，4～16ビット・マイコンでは20年前からあたりまえのものでした．ところが，32ビットRISCプロセッサの世界では，つい最近まであまり省みられていませんでした．ARM社はCortex-M3において，これらの命令への対応を発表しており，ようやく16ビット以下のマイコンの世界に近づいたと言えそうです．

同じようなことはデバッグ機能についても言えます．筆者がARMコアに本格的に取り組んだ当時，すでにリアルタイム・トレースの機能が日本の半導体メーカ各社のマイコンに実装されていました．また，20年前に筆者が設計した8ビット・マイコンでも，すでにこの機能は実現されていました．ところがARMコアの場合，リアルタイム・トレース機能が実装されたのは近年のことであり，電力削減を目指した命令によるスタンバイ・モードが実装されたのも最近になってからのことでした．

これらの違いは，最初のアーキテクチャ設計時のターゲットの市場の違いに起因しているのですが，32ビット組み込みプロセッサもようやく（伝統的な）組み込み機器の市場へ向けて，本格的な対応を始めたと見ることもできそうです．

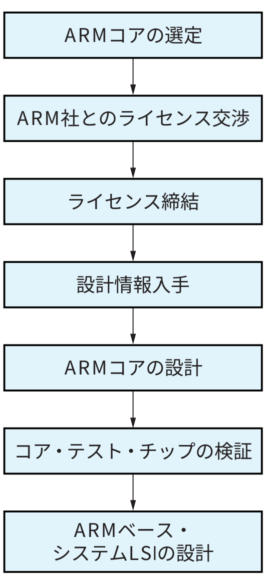


図1　ARMコアの導入から設計までの手順
設計を始める前に，ARM社からARMコアのライセンスを受ける必要がある．

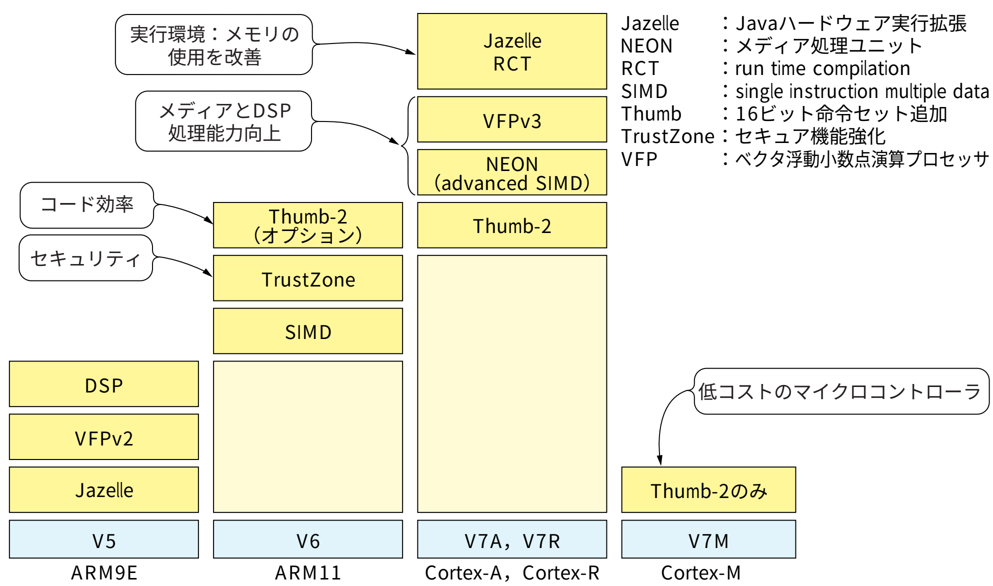


図2　ARMアーキテクチャの進展
Vは命令セット・アーキテクチャのバージョン番号．アーキテクチャ世代の進展に伴って，主要な技術が追加されている．

したThumb2，セキュア機能を強化したTrustZoneなどを搭載している．

- **Cortexファミリ** ── コア・ファミリの命名法が，2004年10月のCortexファミリ・プロセッサ・コア（Cortex-M3）発表時から変更された．新命名法では，プロセッサのアーキテクチャに数字を使わず，名まえを与える．Cortexファミリは，用途ごとに3種類のカテゴリ・グループを形成する．Aグループ（Cortex-A8など）は広い仮想メモリ空間をサポートし，マルチタスク用にMMU（memory management unit）を内蔵する．Rグループ

(Cortex-R4など)はリアルタイム用途を想定して物理メモリ空間だけをサポートしている．メモリを分割して保護するMPU(memory protection unit)を内蔵しており，割り込み機能の強化も図られている．Mグループ(Cortex-M3など)はマイクロコントローラ向け．従来の8ビットや16ビットを中心とするマイクロコントローラの用途をねらっている．

図2にARM9以降のアーキテクチャの強化点を示します．

● ライセンス形態は多様

ARMコアを入手するため，通常はARM社とライセンス契約を結ぶ必要があります．ライセンスにはいくつかの形態があり，おもなものを以下に示します．

- **アーキテクチャ・ライセンス**――コアのマイクロアーキテクチャを変更可能．ただし，既存のARMアーキテクチャとの互換性を維持する必要がある．
- **フルライセンス**――半導体メーカとの間で結ぶことが多い，もっとも一般的な形態．契約したCPUコアの設計・製造・販売の権利を持つことができる．
- **サブスクリプション・ライセンス**――契約で規定された期間に開発されるARMコア・ファミリを自由に使える．
- **シングル・ユース・デザイン・ライセンス(SUDL)**――ただ1回，ある機器製品のためだけに使用でき，設計・製造・販売の権利を持つことができる．特定の機器製品を目的として開発されるLSIに使用できる．おもに機器メーカとの契約を想定している．
- **アカデミック・ライセンス**――大学などの研究機関との契約．CPUコアの種類は限定されるが，ライセンス料はディスカウントされる．

● 無償でARMコアを設計できるプログラム

例えば，本誌の読者が手っ取り早くARMコアにアクセスしたい場合には，DesignStartプログラムを利用すると良いでしょう．ライセンスを得ずに，無償でARMコアを使ったシステムLSIを設計できます．こうした環境が提供されていることは，意外と知られていません．

Webサイトでユーザ登録を行うと，ARM社のユーザ承認を経て，ARM7TDMIのシミュレーション・モデルなどをダウンロードできるようになります．これを使ってシステムLSIを設計できます(実チップが必要になったら，前述のSUDL契約を結んで，実デバイスを製造する権利を得る)．

このプログラムで使用できる設計情報には，以下のものがあります．

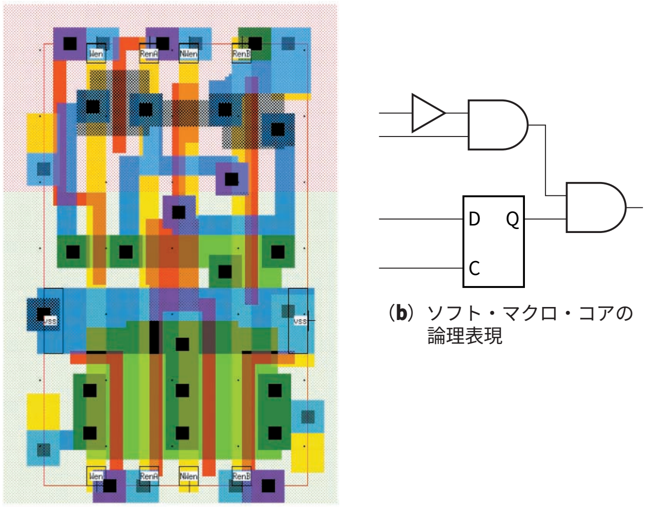
- タイミング情報，フロアプラン情報など(.lib，.FRAM，.LEF，.CLF)
- CPUコアのジェネリックな(汎用的な)RTL(register transfer level)データ，機能限定版のDSM(design sign-off model；サインオフ検証用モデル，詳細は後述)
- 簡易ADK(AMBAバス開発キット)
- 簡易ARM7TDMIバリデーション(システム動作確認用)環境
- テスト・ベクタ(一部を省略したシミュレーション・パターン)
- ドキュメント(実装マニュアル，エラッタ注意書，テクニカル・リファレンス・マニュアル)

● ハード・マクロの移植やソフト・マクロの実装を受託

多くの総合半導体メーカ(IDM：integrated device manufacturer)には，ASIC関連部署とプロセッサ関連部署があります．ARMコアは外部調達のIPコア(LSIに組み込む大規模回路ブロック)としての側面が強いため，伝統的にASIC関連部署が導入を手がけてきました．ただし，ASIC関連部署にはプロセッサの中身や設計を熟知した担当者が少なく，コアの実装に手間取ることもありました．

プロセッサの論理設計や回路設計，レイアウト設計には特有の専門知識と経験が必要です．ARM社はARMコアの実装設計サービスを行っていますが，これがIPベンダとしてのARM社の大きな強みになっていると思います．これはハード・マクロ・コアの開発についてだけでなく，ソフト・マクロ全盛の今日においても同様です．

ここでハード・マクロ・コアとソフト・マクロ・コアの定義を述べておきます．ハード・マクロ・コアは，DSMやレイアウト・レベルのネットリスト，ジェネリックなレイアウト・データの形で提供されます(図3(a))．ライセンシがコアに対して変更を加えることはできません．キャッシュ・サイズなどの構成選択オプションもありません．一方，ソフト・マクロ・コア(論理合成可能コア)は，RTLデータの形で提供され，セル・ライブラリを使って合成できます(図3(b))．ソフト・マクロ・コアについては，ライセンシはキャッシュ・サイズなどの構成変更は可能ですが，機能は変えられません．コアはARMアーキテクチャに準拠していなければなりません．



(a) ハード・マクロ・コアの物理レイアウト(一部)

(b) ソフト・マクロ・コアの論理表現

図3　ハード・マクロ・コアとソフト・マクロ・コア

(a)のハード・マクロ・コアはレイアウト・レベルでライセンスされ，GDS II(マスク・レイアウトのフォーマット)として配布される．ハード・マクロ・コアの例として，ARM710T，ARM920T，ARM1022Eなどがある．(b)のソフト・マクロ・コアはRTLデータとして配布される．ライセンシは，キャッシュ構成を変更可能．機能を変更することはできない．ソフト・マクロ・コアの例として，ARM7TDMI-S，ARM926EJ-S，ARM1136J-Sなどがある．

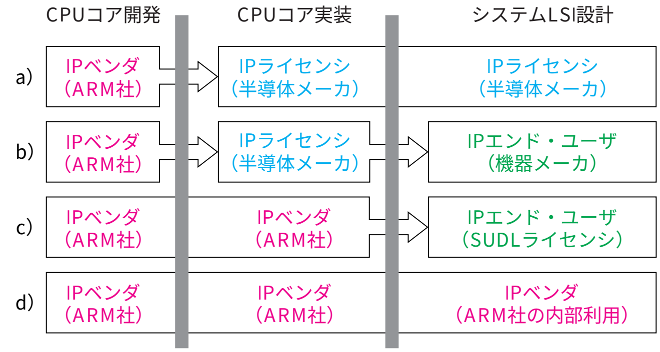


図4　ARMコアのおもな利用・開発形態

IPベンダとIPライセンシ，IPエンド・ユーザの間の関係は，おもに4通りの形態になる．ここで，IPベンダはARM社，IPライセンシは半導体メーカなど，IPエンド・ユーザは機器メーカやファブレスの設計会社になる．

以下に，実装設計サービスの概要を述べます．

1) ハード・マクロ・コアのポーティング(移植)

CPUコアを実装するのはLSI設計のなかでも比較的難しい部類の作業です．その理由としては，高速クロックでコア内部が駆動され，厳しいタイミングで動作していること，また，複雑な制御ブロックを設計する必要があることなどが挙げられます．そこで，ARM社は顧客ごとに異なるCPUのハード・マクロ・コアを設計するポーティング(移植)サービスを行っています．

ARMコアの初期のファミリは，レイアウト・レベルのハード・マクロ・コアとして用意されています．そして，ライセンシ自身が自社の設計ルールに基づいてレイアウトをカスタマイズします．

ARM社はハード・マクロ・コアの汎用的なレイアウトを用意します．そして，ライセンシからの委託を受けて，その会社の設計ルールに合ったハード・マクロ・コアにしあげて提供します．これにより，ライセンシはCPUコアの実装にかかる期間を短縮でき，実チップを入手するまでの時間と開発リソースを抑えることができます．つまり，短い期間でARMコアを搭載したシステムLSIの量産に入れるのです．

2) ソフト・マクロ・コアのハードニング(実装)

論理合成可能なソフト・マクロ・コアとして提供されているCPUコアについては，ARM社はハードニング・サービスと呼ぶ別のサービスを用意しています．これはARM社がライセンシからの委託を受けて，ライセンシのライブラリに基づいてコアの論理合成からレイアウト設計までを行うサービスです．

● ARMコアの利用・開発には四つの形態がある

ARMコアのライセンシは，半導体メーカの場合もあれば，機器メーカの場合もあります．ARMコアの利用・開発の形態は，だいたい以下の4種類に分類できます．

a) ARM社がIPコアを半導体メーカ(ARM社のパートナ企業)へライセンスし，半導体メーカはこれを搭載したチップを自社の顧客に供給する．もっとも一般的な形態．

b) ARM社がIPコアを半導体メーカ(ARM社のパートナ企業)へライセンスし，ライセンシは内部が保護されたIPコアを機器メーカ(OEM企業)に提供する．

c) 機器メーカが直接，ARM社からライセンスを受け，自社でチップを設計して生産・販売する．ARM社は，ファウンドリ・プログラム(特定ファウンドリでARMコアを使ったチップを生産できるプログラム)向けに内部が保護されたIPコアを提供している．

d) ARM社が設計，ハードウェア化し，自社のために使用する．

この分類を図示すると，図4のようになります．

2



● 多くの企業との提携によって成立するIPビジネス

ARMコアを搭載したシステムLSIを開発・利用するには，さまざまな環境・開発支援が必要になります．そのため，ARM社は非常に多くの業態の企業とパートナ契約を結んでいます．以下に，パートナ契約と各社が提供するサービス/製品の例を示します．

● ADC(Approved Design Center)

ADCは，ARMコアを搭載したシステムLSI設計の経験が豊富なARM社認定の設計会社です．国内ではエイ・アイ・エル，大日本LSIデザイン，トッパン・テクニカル・デザインセンター，日本システムウェア，HOYAがあります．これらの企業は設計・生産・販売権利のうちの設計権利を持ち，顧客からの設計依頼に対応します．

● ATC(Approved Training Center)

ATCは，ARMプロセッサに関する技術・開発環境のトレーニングを行う企業です．国内ではアンドールシステムサポートと横河ディジタルコンピュータがあります．

● ALP(ARM Licensing Program)

トッパンは，ARM社とARMコアのサブライセンス・プログラムを締結しています．これにより，特定のARMコアについてのサブライセンス権を得ています．つまり，ARMコアの利用者は，トッパンとARMコアのライセンス契約を結ぶことができます．ただし，コアのライセンス料(契約時に1回だけ支払う料金)とロイヤリティ(チップの数量に応じて支払う料金)は，通常のライセンスの場合と同じように必要です．

● ARMファウンドリ・プログラム

設計が完了すると，基本的には個々の半導体メーカの工場でチップを製造することになりますが，これとは別に海外の半導体ファウンドリ(台湾TSMC，UMCなど)を利用することも可能です．そのため，ARM社はこうしたファウンドリ企業ともファウンドリ・プログラムに基づくパートナ契約を結んでいます．いくつかのARMコアについては，物理モデルや電気的特性などの検証が完了したモデルをファウンドリ企業が用意しています．前述のSUDLライセンスを利用すれば，ファブレスの設計会社も自社ブランドのチップを作ることができます．

● FPGA版のARMコア

米国Actel社は，FPGAに対応したARM7ファミリ・マイクロプロセッサを提供しています．これはARMコア初のFPGA向けソフト・マクロ・コアです．Actel社のフラッシュ・ベースFPGAやアンチヒューズFPGAには，内部のIPコアを保護するセキュリティ機能があります．これにより，FPGA設計者は柔軟性の高いソフト・マクロのARMコアを利用できます．

2. ARMベース・システムLSIの設計手順

契約が完了すると，ARM社から設計に必要なCPUコア・モデル(Verilog HDL，VHDL，その他)やテスト・ベクタ，テクノロジ・リファレンス・マニュアル，ドキュメントなどが送られてきます．ソフト・マクロのARMコアの合成方法は，ARMリファレンス・デザイン・メソドロジというドキュメントに記載されています．

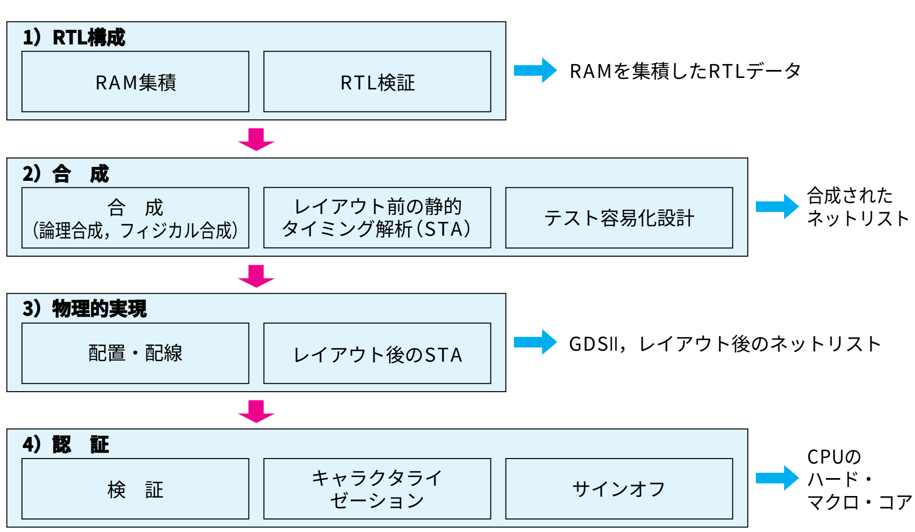


図5
ARMコアの実装フロー
ARMレファレンス・デザイン・メソドロジに基づいた実装フローを示す．このメソドロジでは，論理合成やクロック・ツリーの生成・配置，静的タイミング解析(STA：static timing analysis)などの方法が記述されている．

● コアの開発では，最後に認証の工程がある

最初に，システムLSIに搭載するARMコアそのものの実装を行う必要があります．目標とする条件を満たすARMコアを作るため，ライセンシは事前に以下の項目を確認・確定します．

- 必要なメモリやキャッシュの構成，サイズの設定
- 使用するプロセス技術
- タイミング要求に合うかどうか
- 面積要求に合うかどうか
- 消費電力要求に合うかどうか
- 製造テストのためのテスト容易化設計（例えばスキャン・チェーンの数などを決定する）
- IPコアとしてシステムLSIに組み込むためのブラックボックス・モデル（DSM）の作成

作成されたARMコアに対して，ARM社がIP認証（ARMアーキテクチャ/実装アーキテクチャへの準拠認証）

コア設計を進めるためのフローを**図5**に示します．

1）RTL構成

RTL構成の工程では，CPUコアに対する要求を満たすようにRTLコードを作成します．ここでのおもな作業には，キャッシュ・メモリ用RAMの実装と，キャッシュ・メモリ・テストのためのRAM BIST（built-in self-test）の集積があります．RAMなどの集積が完了したら，RTL検証を行います．

2）合　成

合成の工程では動作環境や性能目標に対して，適切な設計制約を与えなければなりません．例えば，米国Synopsys社のフィジカル合成ツール（Physical Compiler）を使って合成します．論理合成ツール（Design Compilerなど）を使用する場合には，別途，配置工程が必要になります．また，この工程で製造テスト用のスキャン回路を挿入します（最適なスキャン・チェーン設計を行うには，概略配置情報が必要になる）．

3）物理的実現

物理的実現の工程では，配置・配線（レイアウト）と，合成後のタイミング，設計ルール，テスト容易性などのチェックを行います．適切なレイアウト結果が得られるように，フロアプラン設計ガイドラインが用意されています．

4）認証（qualification）

認証の工程では，レイアウト後のネットリストの検証をを行い，ARMコアが正しく実装されたかどうかを確認します．また，最終的なハード・マクロ・コアのキャラクタライゼーション（マクロ・セル用の.libファイルの生成）を行います．さらに，設計の最終認証であるサインオフ・チェックを行います．

● チップ・レベルの検証にはDSMを利用

ARMコアの実装と動作確認が終わると，いよいよARMコアをシステムLSIに組み込む段階になります．**図6**にARMベース・システムLSI設計のフローを示します．これは，通常のシステムLSI設計の手順とまったく変わりません．CPUコアのモデルとしては，DSMを使います．

DSMは各種HDLシミュレーション環境で使用できるモデルで，内部の機能はC言語などで記述されています．また，コアの入出力部（論理・物理境界）のタイミングやコアのアスペクト比（縦横比），ピン配置などが定義されています．コアに入出力する信号のタイミングについては，シミュレーション値や実測値をバックアノテートして反映させることも可能です．

DSMの内部はブラックボックスになっており，詳細な情報は含まれていません．このことは，シミュレーションの高速化に寄与します．ハード・マクロで提供されるコアについては，内部の詳細情報（内部ゲートの構造や回路・レイアウトの実装など）が保護されています．ソフト・マ

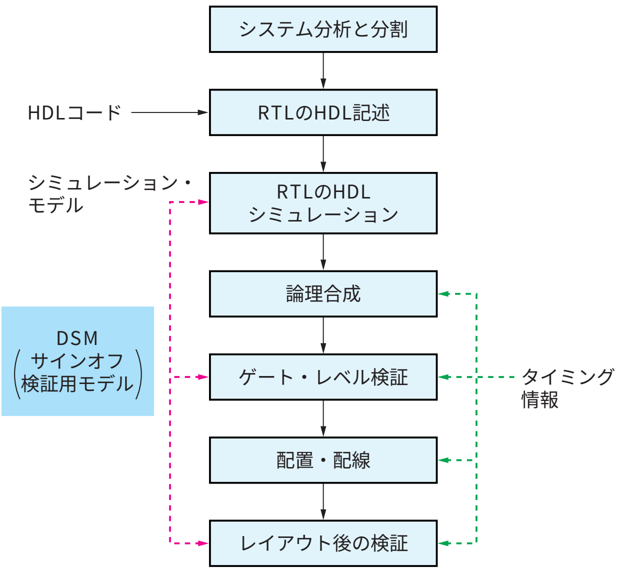


図6　システムLSI設計フロー

ARMコアを搭載したシステムLSIの設計フローは，ほかのシステムLSIの設計フローと同じである．CPUコアの部分については，DSM（サインオフ検証用モデル）を使ってシミュレーションする．

2

クロのコアについても，実装後にDSMで供給される場合は内部情報が保護されます．

システムLSIを設計する際に，DSMをほかのモジュールと組み合わせてシミュレーションできるように，検証環境が整備されています．

3. ESL設計環境を整備

複雑なシステムLSIを設計するうえでは，後工程における手戻りを減らすため，最上流の仕様設計がとても重要になってきています．こうした最上流の設計を支援するための環境として，最近，ESL（electronic system level）設計ツールが注目を集めています．ESL設計ツールを用いると，より抽象度の高い仮想モデルの段階から設計を始められます（**表1**）．

“ESL”についての厳密な定義はなされていないのですが，イメージとしては以下のようなものです．

- ハードウェアとソフトウェアを協調して仮想環境で開発・検証することが可能となる
- 仕様に基づいたシステム・アーキテクチャ実行モデルを作成し，ベンチマーク・プログラムを走らせ，何回もシミュレーションしながら，別のアーキテクチャと比較できる
- 実際のチップができ上がる前に，ソフトウェア開発やOSの移植を始められる

これにより，設計の初期から最終製品のイメージを持って開発作業を進めることができます．早い段階で仕様上の問題点を洗い出せるため，リスクが減り，製品の開発期間を短縮できます（**図7**）．ARM社は，ESL設計ツールとして「RealView SoC Designer（以下，SoC Designer）」という環境をARMベース・システムLSIの開発者に提供しています．

表1　仮想モデルと現実モデル

	仮想モデル	現実モデル
目　的	● アーキテクチャ検討 ● トレードオフ分析 ● IPコア選択の検討 ● シリコン製造前のソフトウェア開発 ● ハードウェアとソフトウェアの最適化	● ハードウェア実装 ● ソフトウェア開発 ● ハードウェア・ソフトウェア協調検証
設計スタイル	ESL（electronic system level）ベース	RTL/シリコン・ベース

● システム・レベルのコンポーネント・ライブラリを用意

SoC Designerは，設計キャンバス（システム・アーキテクチャを入力するためのブロック図エディタ）である「MaxSim Designer」と，シミュレーション環境である「MaxSim Explorer」から構成されます（**図8**）．例えば，サイクル・ベースのSystemCシミュレータと組み合わせてシステム・レベルの仮想プロトタイピング（実行可能な仕様モデルの作成）を行えます．また，マルチコアのプロセッサを開発する際には，複数のCPUコアを同期してシミュレーションすることもできます．

以下に，SoC Designerを使ってシステム・アーキテクチャを定義する際に利用する構成要素（システム・コンポーネント）について説明します．

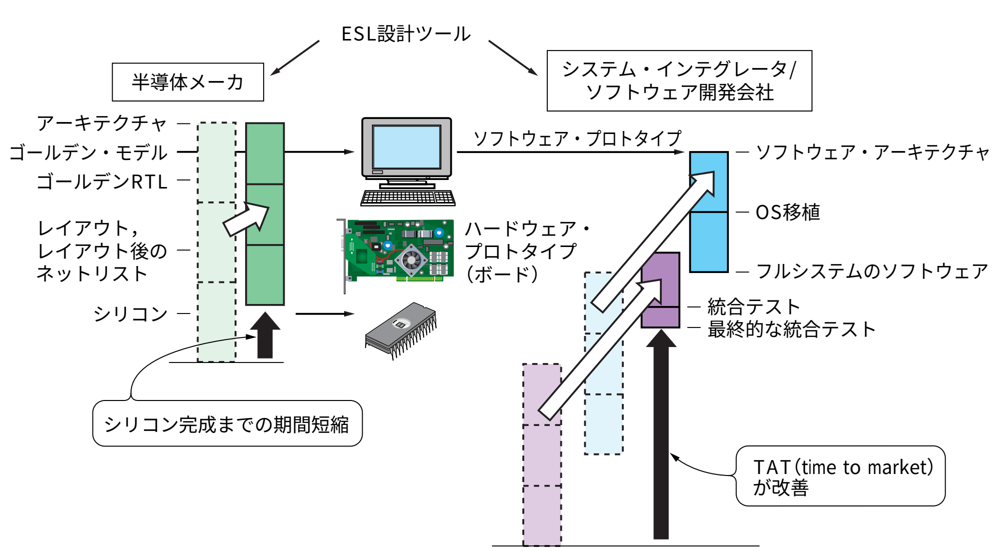
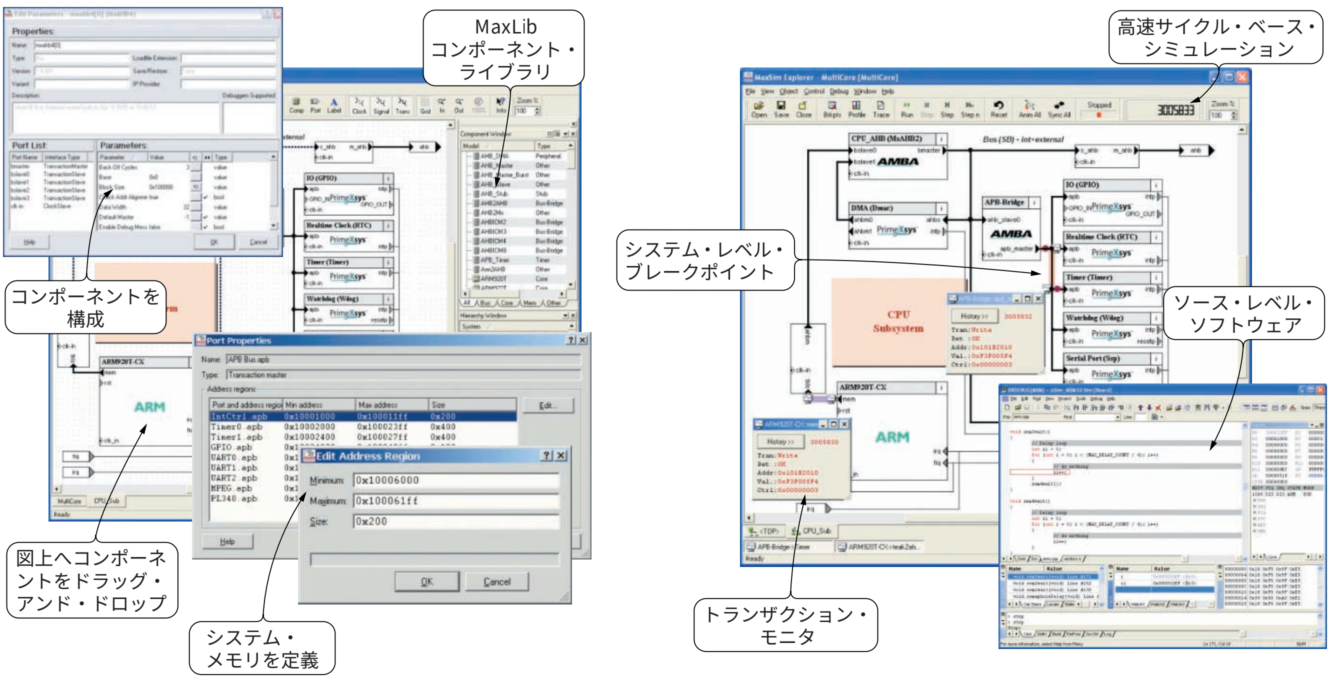


図7
ハードウェアとソフトウェアのコンカレント開発
LSI設計→ボード開発→プログラム開発，というぐあいに順次的に進めていた開発作業を，仮想プロトタイプ（ソフトウェア・プロトタイプ）を用いた並行的な開発に変更する．これにより，全体の開発期間を短縮できる．

1）CPUコア・モデル

ARMコアのシミュレーション・モデルとして，抽象度の高い「リアルタイム・システム・モデル」やタイミング精度の高い「RTLモデル」など，さまざまなモデルが用意されています（**図9**）．例えば，リアルタイム・システム・モデルは実行速度が速く，アプリケーション・ソフトウェアの開発などに利用されます．一方，RTLモデルはLSI設計に利用されます．ユーザは，設計段階に応じて適切なモデルを選択しなければなりません．このうち，SoC Designerで使用するのは，詳細性能の評価やシステム解析などに利用する「サイクル精度（CA）モデル」と，OSブートなどのシミュレーションに用いる「サイクル・アプロキシメート（CX）モデル」です．

SoC Designerのモデル・ライブラリには，ARM社やサ



（**a**）MaxSim Designer　　（**b**）MaxSim Explorer

図8　SoC Designerの画面例

（a）のMaxSim Designerでは，ライブラリのシステム・コンポーネントを組み合わせてシステムを定義する．（b）のMaxSim Explorerでは，定義したシステムの性能を評価したり，ソフトウェアをデバッグする．

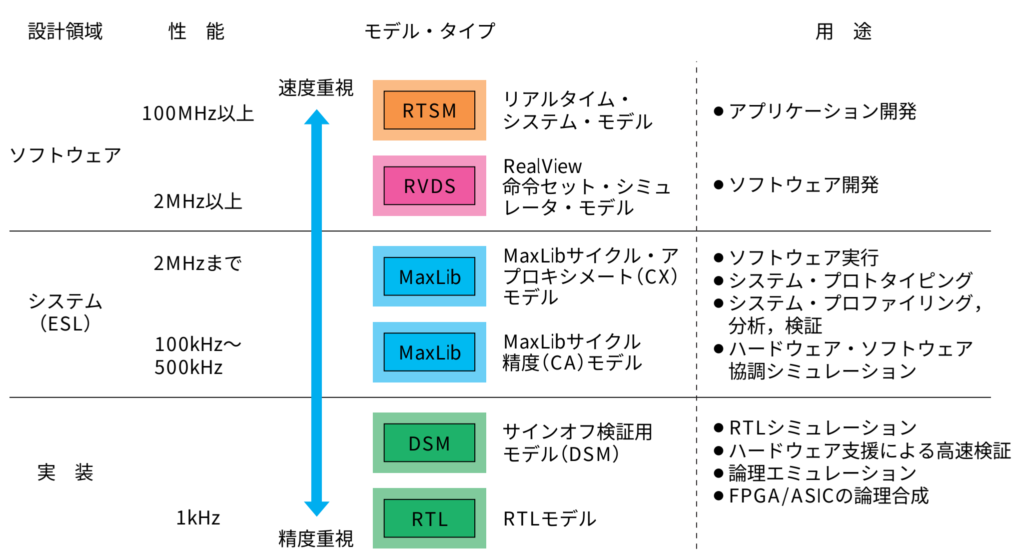


図9
シミュレーション用CPUモデル
開発段階に応じてさまざまなシミュレーション・モデルが用意されている．CPUモデルを適切に選択することによって，開発効率が向上する．

ード・パーティのモデルが格納されています．例えば，ARM7コアからCortex-A8コア，周辺機能モジュール，サード・パーティのプロセッサ・コア（例えば米国CEVA社，米国LSI Logic社，米国MIPS Technologies社，米国StarCore社のコア）などが用意されています．

2）バス構成

システムLSIの設計では，オンチップ・バスをベースにシステムを構築するのが一般的です．SoC Designerにおいても，オンチップ・バスを中心に，CPUコアや周辺機能のコアを接続していきます．オンチップ・バスとして，最近ではAMBA（Advanced Microcontroller Bus Architecture）が広く採用されています．これはARM社が提案した規格で，最新仕様はVersion 3.0となります．AMBAはパケット通信的なバス・プロトコルを採用しています．転送準備ができたデータから順不同でデータ転送を行うことで，バス転送効率を向上させています．

3）周辺機能ライブラリ（MaxLib）

ARM社がライセンシに提供している周辺機能ライブラリとしてPrimeCellライブラリがありますが，これはSoC DesignerのMaxLibライブラリにも含まれています．CPUコアの周辺機能としてキャッシュ・コントローラ，バス・インターコネクト，DMA（direct memory access）コントローラ，メモリ・コントローラなどのモデルが用意されています．

● システム・アーキテクチャのトレードオフ解析が可能

仮想プロトタイピングのため，以上のシステム・コンポーネントをMaxSim Designerの設計キャンバス上に並べ，各コンポーネント間を結線していきます（**図8（a）**）．また，SystemCとC++に対応したコンポーネント・ウィザードが用意されており，これを使ってモデルを生成できます．さらに，SystemCモデルをインポートしたり，エクスポートすることも可能です．

システム・アーキテクチャを定義する手順は，以下のとおりです．

1）設計キャンバス上へ必要なコンポーネントのモデルをドラッグする
2）ブロック図エディタを使ってシステム・コンポーネントを配置し，コンポーネント間を結線する
3）各コンポーネントのパラメータ（バス幅やベース・アドレスなど）を設定する
4）基本ソフトウェア・イメージ（ドライバ，OSなど）をロードする

MaxSim Designerでシステムの構造を定義したら，MaxSim Explorerで性能を確認します（**図8（b）**）．異なるシステム構成（CPUコア，バス・アーキテクチャ，メモリ，キャッシュなど）のトレードオフ解析，デバッグ時のブレークポイントやワッチ・ポイントの設定，ランタイムの可視化，シミュレーション制御，逆アセンブル，ソース・デバッグなどを行えます．

このほか，オンチップ・バスの設計を支援する「AMBA Designer」というツールも用意されています．これにより，オンチップ・バスがシステム・スループットとレイテンシの要求を満たしているかどうかを確認できます．また，MaxSim DesignerやMaxSim Explorerの結果を反映して最適なバス・ファブリック（バス構成）を生成します．

● ソフトウェア環境は新ツールに

現在，ARM社の提供しているARMプロセッサ向けのソフトウェア開発環境にはARM Development Suite（ADS）とRealView Develoment Suite（RVDS）があります．このうちADSは旧世代のツールであり，RVDSに移行しています．RVDSは，コンパイラ，デバッガ，命令セット・シミュレータ（ARMulator），評価ボードから構成されています．

参考・引用*文献

（1）ARM社のWebサイト，http://www.arm.com/，http://www.jp.arm.com/

わたなべ・のぶひさ
アーム（株）　取締役 エンジニアリングディレクター

<著者プロフィール>
渡辺信久．ロボット，からくり人形などの自動機械に興味を持つうちに，初期のマイクロコンピュータと出会う．以来，ゲート数削減のためシフト・レジスタをプログラム・カウンタに使った簡単な4ビット・マイコンや8ビット・マイコン，信号処理プロセッサの開発・設計を手がけ，オンチップ・マルチプロセッサの今日まで，プロセッサ技術の進展とともに歩んできた．今日のように分業化された設計フローでは経験することが難しくなったが，アーキテクチャ設計から論理，回路，セル・レイアウト設計，トランジスタの特性解析まで，幅広く手がけることができたことは幸運だったと感じている．